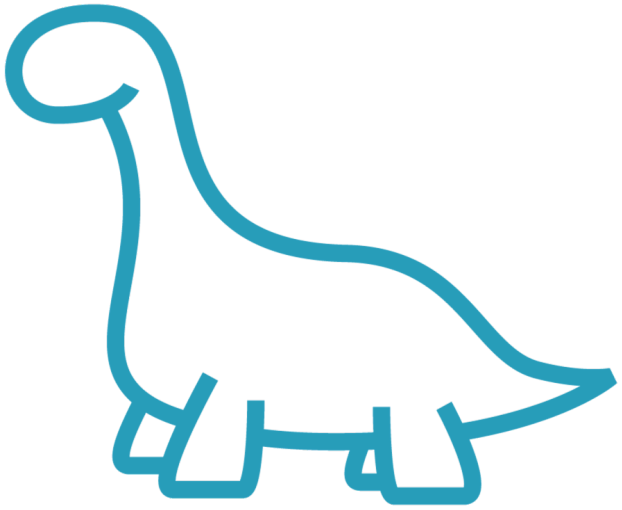
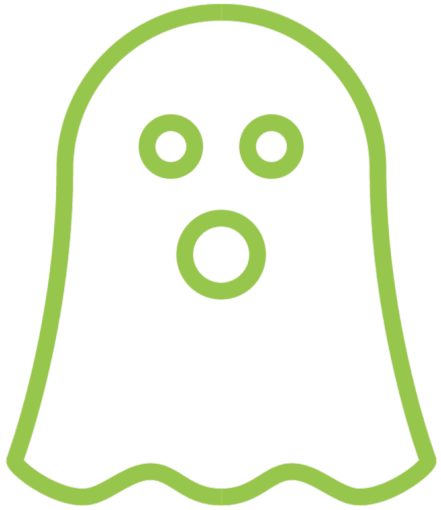
Understanding State Configuration Tools

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

Understanding Configuration Drift



Back in my day, we actually CONFIGURED servers



We used Ghost to reimage hard drives, sometimes dozens of times per day

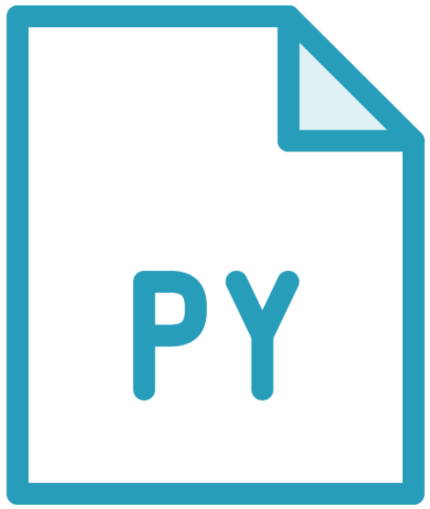
The Drawback of Ghost

Every Ghosting was blowing away what was there

You couldn't do this for a database server, obviously

And images differed for different hardware

Enter Scripting



So, you start scripting this stuff

Powershell

“Install a web server”

The Limitations of Scripting

What if a web server is already there?

You write detection logic

And this gets ever more complex

Idempotence

Having the same effect even happening multiple times.

Idem – the same

Potens - power

Idempotence Scenarios

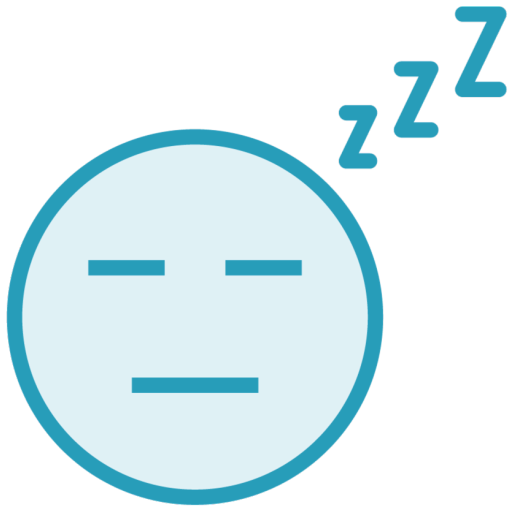
The server has no web server installed

Installed, but with the incorrect configuration

An Unwanted Mail Server

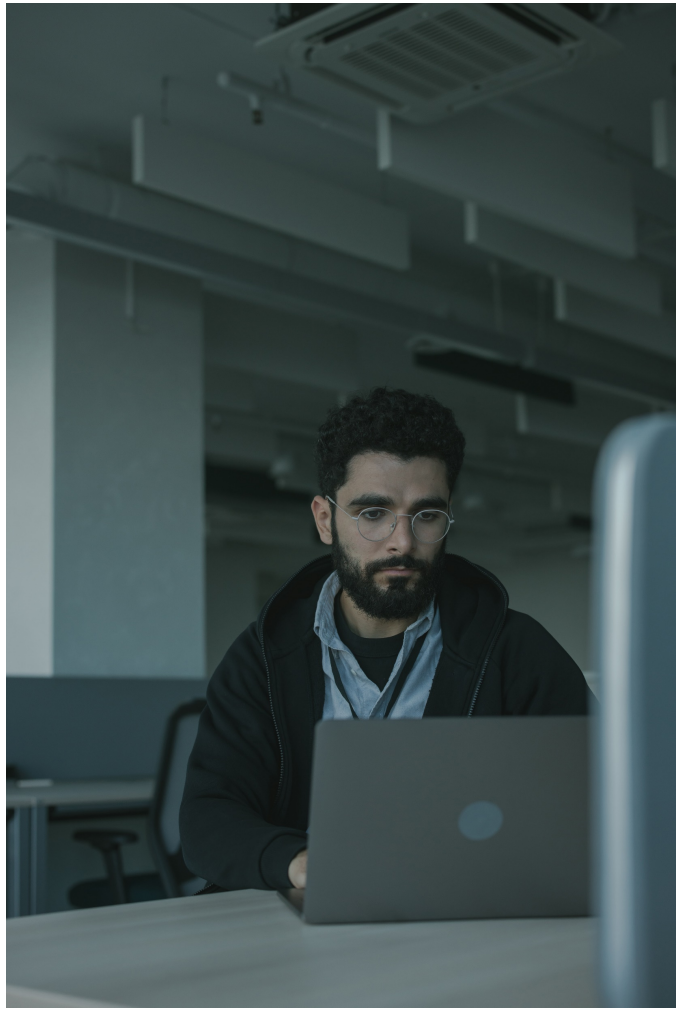
What if there's also an SMTP server?

Attackers would use this to send spam and phishing attacks



Our script won't do anything about this

Manual Configuration



Security usually started out fine

But sometimes people would open ports, permissions and other stuff

Maybe in your enterprise…

- You want to ban TLS 1.0

But 5:00 on Friday comes

- Your IT guy is wanting to leave
- He does something he shouldn’t, intending to fix it later

Maybe he’ll fix it later…maybe

Tools for Mitigating Configuration Drift

Mere execution of scripts and one-time application of state does not mitigate configuration drift

In order to adequately mitigate configuration drift, a tool must enforce configuration state **continually**

The Tools List

Powershell

Infrastructure as Code Orchestrators

- Azure Resource Manager
- Chef
- Puppet
- Ansible
- Terraform

An Introduction to Desired State Configuration in PowerShell

What DSC Is

Our web server script is idempotent

We've figured out all of the implications

What about that rogue mail server?

What about other configuration elements?

A Body of Idempotence Domains

Let's say you figure these all out

Then you can abstract them out of the script

A represent nothing but the script

A DSC Example

```
Configuration WebsiteTest {

        Import-DscResource -ModuleName PsDesiredStateConfiguration

        Node 'localhost' {
                WindowsFeature WebServer {
                        Ensure = "Present"
                        Name   = "Web-Server"
                }
        }
}
```

A DSC Example

```
Configuration WebsiteTest {

        Import-DscResource -ModuleName PsDesiredStateConfiguration

        Node 'localhost' {
                WindowsFeatureSet WindowsFeatureSetExample
            {
                        Name                = @("Web-Ftp-Server", "Web-Server")
                        Ensure              = 'Present'
                        IncludeAllSubFeature    = $true
                }

                WindowsFeature SMTPServer {
                        {
                                Name                = "SMTP-Server"
                                Ensure              = 'Absent'
                        }
                }
        }
}
```

DSC Wrap-up

Primarily oriented for virtual machines

Kubernetes manages this on its own

Principle of least privilege

Course Summary

Release Strategies

- Blue Green
- Rolling
- Canary

Strategies to Minimize Downtime

Branching models for Hotfixes

- A branching model from the ground up
- GitFlow
- A demonstration of what a hotfix looks like to a developer

Configuration drift

- What it is
- Tools to mitigate it
- Desired State Configuration with Powershell